Problem J
Wormholes

A

A friend of yours, an inventor, has built a spaceship recently and wants to explore space with it. During his first voyages, he discovered that the universe is full of wormholes created by some alien race. These wormholes allow one to travel to places far, far away, but moreover, they can also send you to times long ago or in the distant future.
Having mapped these wormholes and their respective end points, you and your friend boldly decide to board his spaceship and go to some distant place you'd like to visit. Of course, you want to arrive at your destination as early as possible. The question is: what is this earliest arrival time?

Input

The first line of input contains an integer c ($1 \leq c \leq 200$), the number of test cases. Each test case starts with a line containing two coordinate triples x_0, y_0, z_0 and x_1, y_1, z_1, the space coordinates of your departure point and destination. The next line contains an integer n ($0 \leq n \leq 50$), the number of wormholes. Then follow n lines, one for each wormhole, with two coordinate triples x_s, y_s, z_s and x_e, y_e, z_e, the space coordinates of the wormhole entry and exit points, respectively, followed by two integers t, d ($-1\,000\,000 \leq t, d \leq 1\,000\,000$), the creation time t of the wormhole and the time shift d when traveling through the wormhole.
All coordinates are integers with absolute values smaller than or equal to 10 000 and no two points are the same.
Note that, initially, the time is zero, and that tunneling through a wormhole happens instantly. For simplicity, the distance between two points is defined as their Euclidean distance (the square root of the sum of the squares of coordinate differences) rounded up to the nearest integer. Your friend's spaceship travels at speed 1.

Output

For each test case, print a single line containing an integer: the earliest time you can arrive at your destination.

Sample Input

```
2
0 0 0 100 0 0
2
1 1 0 1 2 0 -100 -2
0 1 0 100 1 0 -150 10
0 0 0 10 0 0
1
5 0 0 -5 0 0 0 0
```

Sample Output

```
-89
10
```

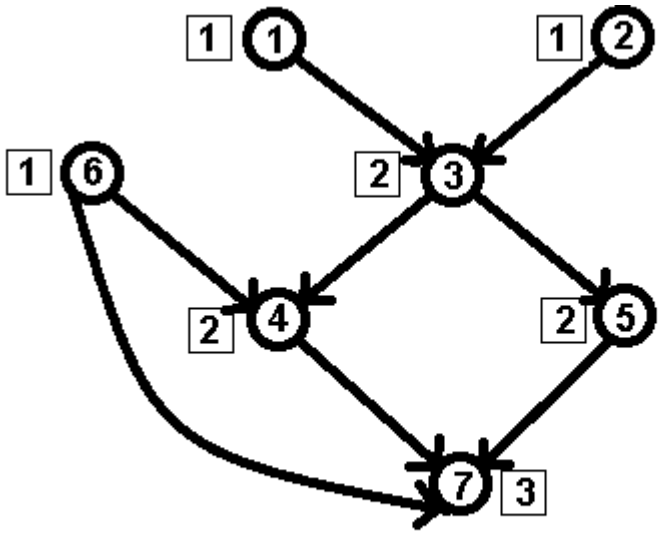
A

This page is intentionally left (almost) blank.

acm 2013
Greater New York Programming Contest
Yale University
New Haven, CT
event sponsors
IBM
YALE

C • Strahler Order

In geology, a river system can be represented as a directed graph. Each river segment is an edge; with the edge pointing the same way the water flows. Nodes are either the source of a river segment (for example, a lake or spring), where river segments merge or diverge, or the mouth of the river.



Note: The number in a box is the order. The number in a circle is the node number.

The *Strahler order* of a river system is computed as follows.

- The order of each source node is `1`.
- For every other node, let `i` be the highest order of all its upstream nodes. If just one upstream node has order `i`, then this node also has order `i`. If two or more upstream nodes have order `i`, then this node has order `i+1`.

The order of the entire river system is the order of the mouth node. In this problem, the river system will have just one mouth. In the picture above, the *Strahler* order is three (3).

You must write a program to determine the order of a given river system.

The actual river with the highest order is the *Amazon*, with order **12**. The highest in the U.S. is the *Mississippi*, with order **10**.

Node **M** is the mouth of the river. It has no outgoing edges.

Input

The first line of input contains a single integer K, ($1 \le K \le 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of multiple lines of input. The first line of each data set contains three positive integers, K, M and P ($2 \le M \le 1000$). K is the data set number. M is the number of nodes in the graph and P is the number of edges. The first line is followed by P lines, each describing an edge of the graph. The line will contain two positive integers, A and B, indicating that water flows from node A to node B ($1 \le A, B \le M$). Node M is the mouth of the river. It has no outgoing edges.

Output

For each data set there is a single line of output. The line consists of the data set number, a single space and the order of the river system.

Sample Input	Sample Output
1 1 7 8 1 3 2 3 6 4 3 4 3 5 6 7 5 7 4 7	1 3

Problem C: Decompressing in a GIF

One well known method to compress image files is the Graphics Interchange Format (GIF) encoding, created by CompuServe in 1987. Here's a simplified version applied to strings of alphabetic characters. Essential for this compression is a dictionary which assigns numeric encodings (we'll use base 10 numbers for this problem) to different strings of characters. The dictionary is initialized with mappings for characters or substrings which may appear in the string. For example, if we expect to encounter all 26 letters of the alphabet, the dictionary will initially store the encodings $(A, 00)$, $(B, 01)$, $(C, 02)$, ..., $(Z, 25)$. If we are compressing DNA data, the dictionary will initially store only 4 entries: $(A, 0)$, $(T, 1)$, $(G, 2)$ and $(C, 3)$. Note that the length of each initial encoding is the same for all entries (2 digits in the first example, and 1 digit in the second).

The compression algorithm proceeds as follows:

1. Find the longest prefix of the uncompressed portion of the string which is in the dictionary, and replace it with its numeric encoding.
2. If the end of the string has not been reached, add a new mapping (s, n) to the dictionary, where $s =$ the prefix just compressed plus the next character after it in the string, and $n =$ the smallest number not yet used in the dictionary.

For example, assume we started with the string `ABABBAABB` and a dictionary with just two entries, $(A, 0)$ and $(B, 1)$. The table below shows the steps in compressing the string.

String	Longest Prefix	Replaced With	New Dictionary Entry
ABABBAABB	A	0	$(AB, 2)$
0BABBAABB	B	1	$(BA, 3)$
01ABBAABB	AB	2	$(ABB, 4)$
012BAABB	BA	3	$(BAA, 5)$
0123ABB	ABB	4	—

The final compressed string is `01234`.

There is only one other rule: the replacement strings used are always the size of the longest encoding in the dictionary at the time the replacement occurs. Thus, with the dictionary above, if the string to compress is long enough that an entry of the form $(s, 10)$ is added to the dictionary, then from this point on all numerical replacement strings used in the compressed string must be expanded to 2 digits long (i.e., `A` will now be encoded as 00, `B` as 01, `AB` as 02, etc.); if an entry $(s', 100)$ is added to the dictionary, all replacements from this point forward will increase to 3 digits long, and so on. Thus, the longer string `ABABBAABBAABAABAB` will be encoded as 01234027301, not 0123402731. Try it!

OK, now that you are experts at compressing, it's time to relax and decompress!

Input

Each test case will consist of two lines. The first line will contain a string of digits to decompress. The second line will contain the initial dictionary used in the compression. This line will start with a positive integer n indicating the number of entries in the dictionary ($1 \leq n \leq 100$), followed by n alphabetic strings. The first of these will be paired with 0 in the dictionary (or 00 if $n > 10$), the second with 1, and so on. The last test case will be followed by a line containing a single 0.

Output

For each test case, output a single line containing the case number (using the format shown below) followed by the decompressed string. All input strings will have been legally compressed.

Sample Input

```
01234
2 A B
01234027301
2 A B
02151120182729
26 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
21104
3 BA A C
01
2 JA VA
0
```

Sample Output

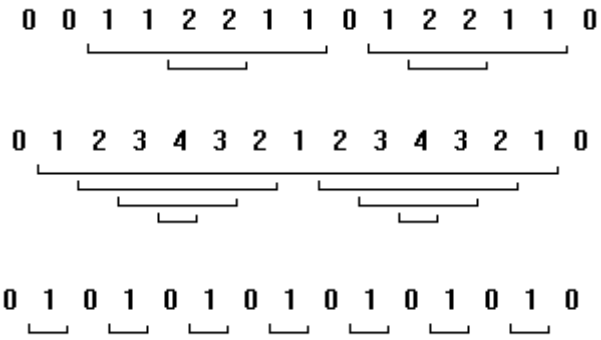
```
Case 1: ABABBAABB
Case 2: ABABBAABBAABAABAB
Case 3: CPLUSPLUS
Case 4: CAABAAA
Case 5: JAVA
```

Greater New York Programming Contest
Yale University
New Haven, CT

event sponsors

A • Islands in the Data Stream

Given a sequence of integers **a1, a2, a3, …, an**, an *island* in the sequence is a contiguous subsequence for which each element is greater than the elements immediately before and after the subsequence. In the examples below, each island in the sequence has a bracket below it. The bracket for an island contained within another island is below the bracket of the containing island.



Write a program that takes as input a sequence of **15** non-negative integers, in which each integer differs from the previous integer by at most **1**, and outputs the number of islands in the sequence.

Input

The first line of input contains a single integer ***P***, (**1** ≤ ***P*** ≤ **1000**), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, ***K,*** followed by **15** non-negative integers separated by a single space. The first and last integers in the sequence will be 0. Each integer will differ from the previous integer by at most 1.

Output

For each data set there is one line of output. The single output line consists of the data set number, ***K***, followed by a single space followed by the number of islands in the sequence.

Sample Input	Sample Output
4 1 0 0 1 1 2 2 1 1 0 1 2 2 1 1 0 2 0 1 2 3 4 3 2 1 2 3 4 3 2 1 0 3 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 4 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0	1 4 2 7 3 7 4 7

D

[D] Probability One

Program:	`guess.(c|cpp|java)`
Input:	`guess.in`
Balloon Color:	`Blue`

Description

Number guessing is a popular game between elementary-school kids. Teachers encourage pupils to play the game as it enhances their arithmetic skills, logical thinking, and following-up simple procedures. We think that, most probably, you too will master in few minutes. Here's one example of how you too can play this game: Ask a friend to think of a number, let's call it n_0. Then:

1. Ask your friend to compute $n_1 = 3 * n_0$ and to tell you if n_1 is even or odd.
2. If n_1 is even, ask your friend to compute $n_2 = n_1/2$. If, otherwise, n_1 was odd then let your friend compute $n_2 = (n_1 + 1)/2$.
3. Now ask your friend to calculate $n_3 = 3 * n_2$.
4. Ask your friend to tell tell you the result of $n_4 = n_3/9$. (n_4 is the quotient of the division operation. In computer lingo, '/' is the integer-division operator.)
5. Now you can simply reveal the original number by calculating $n_0 = 2 * n_4$ if n_1 was even, or $n_0 = 2 * n_4 + 1$ otherwise.

Here's an example that you can follow: If $n_0 = 37$, then $n_1 = 111$ which is odd. Now we can calculate $n_2 = 56$, $n_3 = 168$, and $n_4 = 18$, which is what your friend will tell you. Doing the calculation $2 \times \ n_4 + 1 = 37$ reveals n_0.

Input Format

Your program will be tested on one or more test cases. Each test case is made of a single positive number ($0 < n_0 < 1,000,000$).

The last line of the input file has a single zero (which is not part of the test cases.)

Output Format

For each test case, print the following line:

`k.␣B␣Q`

Where `k` is the test case number (starting at one,) `B` is either `'even'` or `'odd'` (without the quotes) depending on your friend's answer in step 1. `Q` is your friend's answer to step 4.

Sample Input/Output

guess.in

```
37
38
0
```

OUTPUT

```
1. odd 18
2. even 19
```

E

[H] Land Division

Program:	`land.(c|cpp|java)`
Input:	`land.in`
Balloon Color:	Gold

Description

The king of the *Far, Far Away Kingdom* has passed-away and the kingdom must be split amongst his K sons. The kingdom, which can be drawn on a rectangular map, consists of N cities. To divide the land, they will draw $K-1$ straight segments on the map, all of them parallel to either the vertical or the horizontal axis of the map. This divides the map into exactly K rectangles, all having equal heights (if the dividing lines where vertical), or equal widths (if the dividing lines where horizontal). No segment should pass through any of the cities. Each son will then be assigned one random region out of the K regions and the cities inside that region will be his share.

Of course, they want the division to be as fair as possible: theoretically, in the fairest division, each son should get N/K cities (we'll call this value the baseline), but since the baseline isn't always a whole number, each of the sons wants to be as close as possible to the baseline. We will calculate the unfairness of each son as the absolute difference between the number of cities assigned to him and the baseline. The fairest division is the one that minimizes the average unfairness of all the sons.

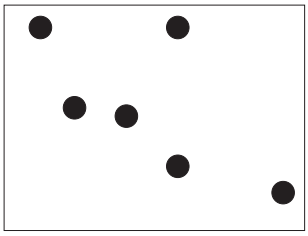


Figure (a)

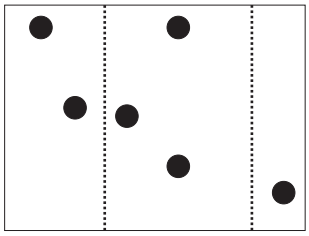


Figure (b)

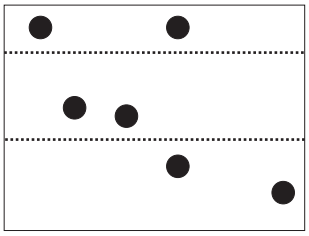


Figure (c)

Consider the example above with 3 sons and 6 cities (so the baseline is $6/3 = 2.0$) Figure (a) is the original map. Figure (b) shows a non-optimal division (the dashed lines are the 2 dividing lines.) In this case, the middle region contains 3 cities (unfairness of $|3-2| = 1$), the left region contains 1 city (unfairness of $|1-2| = 1$), while the right region contains 2 cities (perfectly fair, unfairness of 0), so the average unfairness is $2/3$.

Figure (c) on the right shows the optimal division since all three regions contain the same number of cities for an average unfairness of 0.

Write a program to determine the fairest land division for a given kingdom.

F

Input Format

Your program will be tested on one or more test cases. Each test case is described on $N+1$ lines. The first line of each test case specifies two positive integers: ($N \leq 100,000$) and ($K \leq 10$) where N is the number of cities and K is number of children. Note that $K \leq N$.

N lines follows, each describing the coordinates of a city by specifying two integers (x, y) where $0 \leq x, y \leq 100,000$. Since coordinates are rounded to the nearest integer, more than one city could have the exact same coordinate on the map. You may assume that the map of the kingdom is any rectangle that contains all of the given points (although such information is not needed by the program.) Note also that while all cities lie on integer coordinates, the dividing lines need not be.

The last line of the input file contains two zeros.

Output Format

For each test case, print the following line:

```
k.␣A/B
```

Where `k` is the test case number (starting at one,) and `A/B` is the minimum average that could be obtained. `A/B` should be an irreducible fraction. Let `B=1` when the result is a whole number.

Sample Input/Output

land.in

```
6 3
0 4
1 3
2 3
3 1
4 4
5 0
4 3
0 0
0 1
1 1
1 0
0 0
```

OUTPUT

```
1. 0/1
2. 8/9
```

acm 2006
Greater New York Programming Contest
Nassau Community College
Garden City, NY
event sponsors
IBM
AdaCore The GNAT Pro Company
Google

A • Quick Change

J.P. Flathead's Grocery Store hires cheap labor to man the checkout stations. The people he hires (usually high school kids) often make mistakes making change for the customers. Flathead, who's a bit of a tightwad, figures he loses more money from these mistakes than he makes; that is, the employees tend to give more change to the customers than they should get.

Flathead wants you to write a program that calculates the number of quarters ($0.25), dimes ($0.10), nickels ($0.05) and pennies ($0.01) that the customer should get back. Flathead always wants to give the customer's change in coins if the amount due back is $5.00 or under. He also wants to give the customers back the smallest total number of coins. For example, if the change due back is $1.24, the customer should receive 4 quarters, 2 dimes, 0 nickels, and 4 pennies.

Input

The first line of input contains an integer ***N*** which is the number of datasets that follow. Each dataset consists of a single line containing a single integer which is the change due in cents, **C**, ($1 \leq \mathbf{C} \leq 500$).

Output

For each dataset, print out the dataset number, a space, and the string:

```
Q QUARTER(S), D DIME(S), n NICKEL(S), P PENNY(S)
```

Where ***Q*** is he number of quarters, ***D*** is the number of dimes, ***n*** is the number of nickels and ***P*** is the number of pennies.

Sample Input	Sample Output
3 124 25 194	1 4 QUARTER(S), 2 DIME(S), 0 NICKEL(S), 4 PENNY(S) 2 1 QUARTER(S), 0 DIME(S), 0 NICKEL(S), 0 PENNY(S) 3 7 QUARTER(S), 1 DIME(S), 1 NICKEL(S), 4 PENNY(S)

G

Problem F: Here's a Product Which Will Make You Tensor

Most people are familiar with how to multiply two matrices together. However, an alternate form of multiplication known as tensor multiplication exists as well, and works more like you would expect matrix multiplication should. Let A be a $p \times q$ matrix and B be an $n \times m$ matrix, where neither A nor B is a 1×1 matrix. Then the tensor product $A \otimes B$ is a $pn \times qm$ matrix formed by replacing each element a_{ij} in A with the matrix $(a_{ij}) \cdot B$. Two examples are shown below, which also demonstrate that, like normal matrix multiplication, tensor multiplication is non-commutative:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 4 \\ 3 & 3 & 3 & 4 & 4 & 4 \\ 3 & 3 & 3 & 4 & 4 & 4 \\ 3 & 3 & 6 & 4 & 4 & 8 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 & 3 & 4 \\ 1 & 2 & 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 & 3 & 4 \\ 1 & 2 & 1 & 2 & 2 & 4 \\ 3 & 4 & 3 & 4 & 6 & 8 \end{bmatrix}$$

Note that there is no restriction that the number of columns in the first matrix must equal the number of rows in the second, as there is with normal matrix multiplication. The object of this problem is to determine the number of ways (if any) a given matrix can be formed as a result of a tensor multiplication.

Input

The first line of input for a test case will contain two positive integers r and c indicating the number of rows and columns in the matrix. After this will follow r lines each containing c positive integers. The values of r and c will be ≤ 500, each entry in the matrix will be no greater than 65,536, and the last test case is followed by a line containing `0 0`.

Output

For each test case, output the number of different ways the matrix could be the tensor product of two positive integer matrices, neither of which is a 1×1 matrix.

Sample Input

```
6 6
1 1 1 2 2 2
1 1 1 2 2 2
1 1 2 2 2 4
3 3 3 4 4 4
3 3 3 4 4 4
3 3 6 4 4 8
2 2
3 6
4 9
2 4
15 18 30 36
20 24 40 48
0 0
```

Sample Output

```
1
0
4
```

H

Problem G
Room Assignments

I

Once there was an inventor congress, where inventors from all over the world met in one place. The organizer of the congress reserved exactly one hotel room for each inventor. Each inventor, however, had its own preference regarding which room he would like to stay in. Being a clever inventor himself, the organizer soon found an objective way of doing the room assignments in a fair manner: each inventor wrote two different room numbers on a fair coin, one room number on each side. Then, each inventor threw his coin and was assigned the room number which was shown on the upper side of his coin. If some room had been assigned to more than one inventor, all inventors had to throw their coins again.

As you can imagine, this assignment process could take a long time or even not terminate at all. It has the advantage, however, that among all possible room assignments, one assignment is chosen randomly according to a uniform distribution. In order to apply this method in modern days, you should write a program which helps the organizer.

The organizer himself needs a hotel room too. As the organizer, he wants to have some advantage: he should be able to rate each of the rooms (the higher the rating, the better), and the program should tell him which two room numbers he should write on his coin in order to maximize the expected rating of the room he will be assigned to. The program also has access to the choices of the other inventors before making the proposal. It should never propose two rooms for the organizer such that it is not possible to assign all inventors to the rooms, if a valid assignment is possible at all.

Input

The input starts with a single number c ($1 \leq c \leq 200$) on one line, the number of test cases. Each test case starts with one line containing a number n ($2 \leq n \leq 50\,000$), the number of inventors and rooms. The following $n-1$ lines contain the choices of the $n-1$ guests (excluding the organizer). For each inventor, there is a line containing two numbers a and b ($1 \leq a < b \leq n$), the two room numbers which are selected by the inventor. The last line of each test case consists of n integers $v_1, \ldots, v_n$ ($1 \leq v_i \leq 1\,000\,000$), where v_i is the organizer's rating for room i.

Output

For each test case, print a single line containing the two different room numbers a and b which should be selected by the organizer in order to maximize the expected rating of the room he will be assigned to. If there is more than one optimal selection, break ties by choosing the smallest a and, for equal a, the smallest b. If there is no way for the organizer to select two rooms such that an assignment of inventors to rooms is possible, print "`impossible`" instead.

Sample Input

```
3
4
1 2
2 3
1 3
2 3 4 1
3
1 2
2 3
100 40 70
5
1 2
1 2
1 2
3 4
1 1 1 1 1
```

Sample Output

```
1 4
1 3
impossible
```

I

F: Fred's Lotto Tickets

Fred likes to play the lotto. Whenever he does, he buys lots of tickets. Each ticket has 6 unique numbers in the range from **1** to **49**, inclusive. Fred likes to "Cover all his bases." By that, he means that he likes for each set of lottery tickets to contain every number from **1** to **49**, at least once, on some ticket. Write a program to help Fred see if his tickets "Cover all the bases."

Input

The input file consists of a number of test cases. Each case starts with an integer ***N*** (**1 <= *N* <= 100**), indicating the number of tickets Fred has purchased. On the next ***N*** lines are the tickets, one per line. Each ticket will have exactly 6 integers, and all of them will be in the range from **1** to **49** inclusive. No ticket will have duplicate numbers, but the numbers on a ticket may appear in any order. The input ends with a line containing only a **0**.

Output

Print a list of responses for the input sets, one per line. Print the word **Yes** if every number from **1** to **49** inclusive appears in some lottery ticket in the set, and **No** otherwise. Print these words exactly as they are shown. Do not print any blank lines between outputs.

Sample Input

```
1
1 2 3 4 5 6
9
1 2 3 4 5 6
10 9 8 7 12 11
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
31 32 33 34 35 36
37 38 39 40 41 42
43 44 45 46 47 48
49 19 34 27 25 13
0
```

Sample Output

```
No
Yes
```

J

Greater New York Programming Contest
Yale University
New Haven, CT

event sponsors

D • Pisano Periods

In 1960, Donald Wall of IBM, in White Plains, NY, proved that the series obtained by taking each element of the *Fibonacci* series modulo `m` was periodic.

For example, the first ten elements of the *Fibonacci* sequence, as well as their remainders modulo `11`, are:

```
n            | 1  2  3  4  5  6   7   8   9  10
F(n)         | 1  1  2  3  5  8  13  21  34  55
F(n) mod 11  | 1  1  2  3  5  8   2  10   1   0
```

The sequence made up of the remainders then repeats. Let `k(m)` be the length of the repeating subsequence; in this example, we see `k(11) = 10`.

Wall proved several other properties, some of which you may find interesting:

- If `m > 2`, `k(m)` is even.
- For any even integer `n > 2`, there exists m such that `k(m) = n`.
- $k(m) \leq m^2 - 1$
- $k(2^n) = 3 * 2^{(n-1)}$
- $k(5^n) = 4 * 5^n$
- $k(2 * 5^n) = 6n$
- If $n > 2$, $k(10^n) = 15 * 10^{(n-1)}$

For this problem, you must write a program that calculates the length of the repeating subsequence, `k(m)`, for different modulo values `m`.

Input

The first line of input contains a single integer ***P***, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set is a single line that consists of two space separated integer values ***N*** and ***M***. ***N*** is the data set number. ***M*** is the modulo value (`2 <= m <= 1,000,000`).

Output

For each data set there is one line of output. It contains the data set number (***N***) followed by a single space, followed by the length of the repeating subsequence for ***M, k(M)***.

Sample Input	Sample Output
5 1 4 2 5 3 11 4 123456 5 987654	1 6 2 20 3 10 4 15456 5 332808

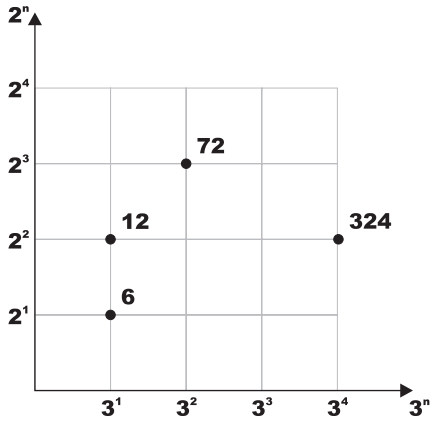
[C] Not So Flat After All

Program:	`space.(c|cpp|java)`
Input:	`space.in`
Balloon Color:	Green

Description

Any positive integer v can be written as $p_1^{a_1} * p_2^{a_2} * \ldots * p_n^{a_n}$ where p_i is a prime number and $a_i \geq 0$. For example: $24 = 2^3 * 3^1$.

Pick any two prime numbers p_1 and p_2 where $p_1 \neq p_2$. Imagine a two dimensional plane where the powers of p_1 are plotted on the x-axis and the powers of p_2 on the y-axis. Now any number that can be written as $p_1^{a_1} * p_2^{a_2}$ can be plotted on this plane at location $(x, y) = (a_1, a_2)$. The figure on the right shows few examples where $p_1 = 3$ and $p_2 = 2$.

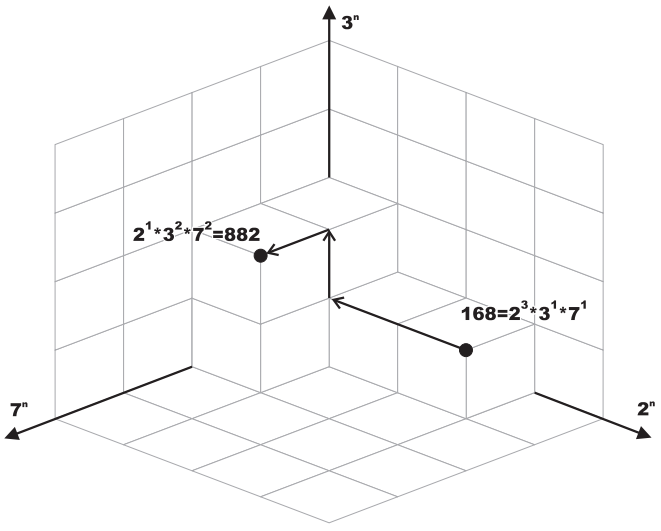


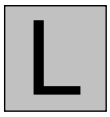
This idea can be extended for any N-Dimensional space where each of the N axes is assigned a unique prime number. Each N-Dimensional space has a unique set of primes. We call such set the *Space Identification Set* or S for short. $|S|$ (the ordinal of S) is N.

Any number that can be expressed as a multiplication of $p_i \in S$ (each raised to a power $(a_i \geq 0)$ can be plotted in this $|S|$-Dimensional space. The figure at the bottom illustrates this idea for $N = 3$ and $S = \{2, 3, 7\}$. Needless to say, any number that can be plotted on space `A` can also be plotted on space `B` as long as $S_A \subset S_B$.

We define the distance between any two points in a given N-Dimensional space to be the sum of units traveled to get from one point to the other while following the grid lines (i.e. movement is always parallel to one of the axes.) For example, in the figure below, the distance between 168 and 882 is 4.

Given two positive integers, write a program that determines the minimum ordinal of a space where both numbers can be plotted in. The program also determines the distance between these two integers in that space.





Input Format

Your program will be tested on one or more test cases. Each test case is specified on a line with two positive integers ($0 < A, B < 1,000,000$) where $A * B > 1$.

The last line is made of two zeros.

Output Format

For each test case, print the following line:

`k.␣X:D`

Where `k` is the test case number (starting at one,) `X` is the minimum ordinal needed in a space that both A and B can be plotted in. `D` is the distance between these two points.

Sample Input/Output

space.in

```
168 882
770 792
0 0
```

OUTPUT

```
1. 3:4
2. 5:6
```